Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

Benchmarking stiff ODE solvers for atmospheric chemistry problems I: implicit versus explicit

A. Sandu, J.G. Verwer, M. van Loon, G.R. Carmichael,
F.A.Potra, D. Dabdub and J.H. Seinfeld

Department of Numerical Mathematics

NM-R9603 February 29, 1996

Report NM-R9603
ISSN 0169-0388

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Benchmarking Stiff ODE Solvers for Atmospheric Chemistry Problems I: Implicit versus Explicit

A. Sandu
Program in Applied Mathematics and Computational Sciences
The University of Iowa, Iowa City, IA 52246, USA

J.G. Verwer, M. van Loon
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

G.R. Carmichael
Center for Global and Regional Environmental Research
The University of Iowa, Iowa City, IA 52246, USA

F.A. Potra
Departments of Mathematics and Computer Science
The University of Iowa, Iowa City, IA 52246, USA

D. Dabdub
Department of Mechanical and Aerospace Engineering
University of California Irvine, Irvine, CA 92717-3975, USA

J.H. Seinfeld
Department of Chemical Engineering
California Institute of Technology, Pasadena, CA 91125, USA

Abstract

In many applications of atmospheric transport-chemistry problems, a major task is the numerical integration of the stiff systems of ordinary differential equations describing the chemical transformations. This paper presents a comprehensive numerical comparison between five explicit and four implicit solvers for a set of seven benchmark problems from actual applications. The results presented may constitute a guide for atmospheric modelers to select a suitable integrator based on the type and dimension of their chemical mechanism and on the desired level of accuracy. Furthermore, we would like to consider this paper an open invitation for other groups to add new representative test problems to those described here and to benchmark their numerical algorithms in our standard computational environment.

AMS Subject Classification (1991): Primary: 65L05. Secondary: 80A30, 65F05.
CR Subject Classification (1991): G1.7
Keywords & Phrases: Atmospheric chemistry, Air pollution modeling, Numerical stiff ODEs.
Note: This work has also been released in the report series 'Reports on Computational Mathematics' of the University of Iowa (1996). For further publication the manuscript has been submitted to the journal Atmospheric Environment.

1. Introduction

To better understand the transport and fate of trace gases and pollutants in the atmosphere, comprehensive atmospheric transport-chemistry models have been developed. For their numerical solution,

very often the operator splitting approach is followed. A major computational task is then the numerical integration of the stiff ODE (ordinary differential equation) systems describing the chemical transformations. These systems take the special production-loss form

$$\frac{dy}{dt} = f(t,y) := P(t,y) - L(t,y)\,y, \qquad y(t) = (y_1(t), \ldots, y_m(t))^T, \tag{1.1}$$

where $L(t,y)$ is a diagonal matrix. The integration must be carried out repeatedly at all spatial grid points for all split intervals chosen. Consequently, a full transport-chemistry computation involves a huge number of stiff ODE integrations. Although the numerical stiff ODE field is well developed and a variety of efficient and quite reliable stiff ODE solvers is available [11], the general experience is that still faster solvers are needed. There are many such solvers in use in atmospheric models. Often the numerical methods are intertwined with the chemistry scheme, like in QSSA methods [12, 13, 26]. In such a case a change in the chemistry scheme necessitates a reconsideration of the numerical scheme as well, which is a disadvantage. Furthermore, this intertwinement makes it very hard to really assess and compare the numerical efficiency and accuracy for different solvers. Continuing our previous work (see [16, 25] and the references therein), the purpose of the current paper is a comprehensive benchmark comparison between a number of solvers which have been proposed in the literature. We apply them as normal ODE solvers without any intertwinement with the chemistry scheme.

The paper is organized as follows. In Section 2 we briefly describe our test set. This set consists of seven problems based on three tropospheric gas-phase chemistry schemes, namely a small, a medium and a large scheme, one stratospheric scheme coming from NASA, and one hybrid gas-liquid phase scheme from cloud modeling which we obtained from Matthijsen [18]. Because of the diverse applications, these chemical schemes constitute a representative test set for evaluating and comparing numerical solvers. Section 3 is devoted to the solvers. We have tested nine existing solvers, namely TWOSTEP [24, 25, 26], CHEMEQ [29], the most simple QSSA solver, an extrapolated QSSA solver [16], the QSSA solver ET based on the extrapolation technique of [4], as well as the implicit solvers LSODES [14], VODE [2], SDIRK4 [11] and RODAS[1] [11].

The first five of these are special purpose and compute the solution in an explicit way, as opposed to the last four which are all implicit and state-of-the-art in the numerical stiff ODE field. For our purpose, the solvers VODE, SDIRK4 and RODAS have been provided with a sparse-matrix technique [19] to economize on the numerical algebra overhead in the modified Newton solution. LSODES has already built in a sparse-matrix technique and is a successor of LSODE [14] which is popular amongst atmospheric chemists as a reference code. By providing the implicit solvers with sparse matrix techniques, they belong to the fastests of their kind. In Section 4 we describe the set up of the experiments. Section 5 contains the results of the comparisons and the final Section 6 collects a number of general remarks and final conclusions.

To enable interested readers to further extend this benchmark comparison using their own solvers, as well as to extend our problem set with other challenging example problems from atmospheric chemistry, all the software we have used for the problems and the solvers has been put on a ftp-site (see [9] and the instructions therein).

2. The benchmark problems

The seven test problems are based on a set of five chemical schemes which are presently being used in various studies. Four of them describe gas-phase, and one describes gas-liquid phase chemistry. All are fully documented elsewhere. Before briefly describing each problem, several general remarks are in order:

[1] RODAS is a Runge-Kutta-Rosenbrock solver and hence not implicit in the strict mathematical sense. However, a Rosenbrock method requires the solution of linear systems of algebraic equations, like an implicit method does if the iterative Newton technique is used. For ease of presentation, in this paper we therefore also call RODAS implicit.

- All the test problems were uniformly coded in FORTRAN using the KPP symbolic preprocessor [5]. This uniformity is important for a meaningfull intercomparison, since part of the algorithms need the derivative function in production-destruction form, part need it in the standard form, and some of them need an analytical Jacobian. None of the solvers was favoured/inhibited by a cheaper/ more expensive implementation of these functions. FORTRAN code defining the test problems can be obtained from [9].

- All problems were run for five days. This time interval is sufficiently large for taking into account several diurnal cycles originating from the photochemical reactions. For all models the unit of components of dy/dt used in the numerical tests is number of molecules/cm^3/second.

- The tropospheric gas-phase problems are based on three different chemical schemes. These are the 15-species EUSMOG scheme, the 32-species CBM-IV scheme, and the 84-species mechanism implemented in the STEM-II model. This mechanism will be referred to as AL. All three are used in present applications and are representative of those being used in the atmospheric chemistry models. Varying the size of the mechanism is important since both implicit and explicit solvers are considered for this benchmark. The stratospheric gas-phase problem contains 34 species and the tropospheric gas-liquid phase one 65.

- The same urban and rural scenarios are simulated with CBM-IV and with AL. Although the chemical conditions are identical and the calculated results very close, the performances of the numerical solvers depend on the chemical mechanism used. We will make this point later in the paper.

- An important issue in our numerical comparison is the use of a sparse matrix technique [19] to economize on the linear algebra costs which the stiff solvers spend in the modified Newton iteration. As a measure of these costs, we give in Table 1 the number of nonzero elements in the Jacobian matrix, as well as the number of nonzero entries in the Newton matrix after the LU factorization. The ratio between the number of nonzeroes in the Jacobian matrix and the square of the dimension gives an indication to which extent a sparse matrix solution may improve the timing compared to a standard dense matrix solution. If this ratio is small, say less than about 1/4, and a reordering of the species exists which gives rise to a small fill-in after the factorization, then a good sparse solution technique will be significantly more efficient than the standard dense solution. The table shows that for our test problems both the ratio for the Jacobian and the resulting fill-in are quite small. The sparse matrix technique we have used is based on a diagonal Markowitz criterion (see Section 3.5 for some more details). Lest we miss the obvious, for problems of a small dimension for which the dense matrix numerical algebra costs are not dominating, the gain in using a more efficient matrix solution will be hardly noticeable in the overall costs. This is the case for Problem `A` (see also [25]).

- The special purpose explicit algorithms exploit the production-loss form of the ODE system and are based on the assumption that all short lived species, causing the problem to be stiff, are only weakly coupled to all other species. In mathematical terms this implies that for these short lived species the loss term $-L_i(t,y)$ is close to a (stiff) eigenvalue of the Jacobian matrix, and that no stiff eigenvalues exist which are not close to a loss term. Although this interpretation is not mathematically rigorous for the nonlinear problems we deal with, it predicts to a great extent whether an explicit solver of the type used in this paper can be justified in advance[2]. For each problem we therefore illustrate this relationship in a table showing the species and eigenvalues for which the relationship is found to exist and the distribution of the remaining part of the spectrum [3] of the Jacobian. In this remaining part, eigenvalues thus can be of two sorts, either

[2] For one of these, namely TWOSTEP using Gauss-Seidel iteration, the coupling between short and long lived species may be stronger, since Gauss-Seidel iteration introduces a form of 'triangular implicitness'.

[3] All the eigenvalues were computed with the routine `dgees` from LAPACK.

Problem	A	B, C	D, E	F	G
Dimension	15	32	84	34	65
Jacobian	57	276	674	246	506
Factorized	57	300	768	280	629

Table 1: The dimension of the test problems, the number of nonzeroes in the Jacobian matrix, and the number of nonzeroes in the Newton matrix after the LU factorization. The difference between the numbers in the third and second row is the fill-in.

Problem	A
Species i	$\mathrm{Re}(\lambda_i)$
OH	-9
All other $\in$	$[-4 \cdot 10^{-4},\ +2.5 \cdot 10^{-5}]$

Table 2: Distribution of real part of the spectrum of the Jacobian for EUSMOG problem A.

they are small and hence do not introduce stiffness, or they are large but cannot be associated with a single short lived species. If these latter eigenvalues exist, then the special purpose explicit methods can fail completely. Inspection of all the tables presented here will reveal that these latter eigenvalues exist only for the tropospheric wet problem G. This observation is in line with our test results.

2.1 Problem A*: The* EUSMOG *model*

This problem is borrowed from a model which is currently implemented and tested at the CWI in a Dutch smog prediction code in the framework of the project EUSMOG [22, 23]. The problem is a highly parameterized version of the EMEP MSW-W ozone chemistry scheme [21]. It consists of 15 reactions between 15 species and is extensively described in [22]. It has been used before in the comparisons reported in [25], where it has also been documented. Information about the eigenvalues can be found in Table 2.

2.2 Problems B *and* C*: The* CBM-IV *model*

These problems are based on the Carbon Bond Mechanism IV (CBM-IV) (Gery et al., [10]), consisting of 32 chemical species involved in 70 thermal and 11 photolytic reactions. The concentration of H_2O was held fixed throughout simulation. The CBM-IV mechanism was designed for the numerical simulation of chemical processes in urban and in regional scale models. Test problem B describes an urban scenario and simulates a heavily polluted atmosphere. The initial concentrations and the levels of hourly emissions follow those described in [20]. This helps to relate our results to those presented in the above mentioned paper. Test problem C describes a rural scenario and simulates a clean atmosphere. It follows the IPCC Chemistry Intercomparison Study, third Bio scenario (see [8]). The values of initial concentrations and the values of hourly emissions are given in Table 5. The emission was released in equal quantities at the beginning of each time interval. Information about the stiffness of the models in terms of the eigenvalues of the Jacobian is presented in Table 3.

2.3 Problems D *and* E*: The* AL *model*

The test problems D and E are based on the largest chemical system tested here. They employ the kinetic mechanism that is presently used in the STEM-II regional-scale/transport/chemistry/removal model (Carmichael et al., [3]), consisting of 84 chemical species[4] involved in 142 thermal and 36

[4]Plus 4 species whose concentrations were held fixed throughout simulation: H_2O, CO_2, O_2, H_2.

Problem	B (urban)	C (rural)
Species i	$\mathrm{Re}(\lambda_i)$	$\mathrm{Re}(\lambda_i)$
$O(^1D)$	$-8.11 \cdot 10^8$	$-8.11 \cdot 10^8$
$O(^3P)$	$-8.26 \cdot 10^4$	$-8.26 \cdot 10^4$
ROR	$-2.47 \cdot 10^3$	$-2.46 \cdot 10^3$
OH	-46	-3.5
TO_2	-4.27	-4.2
All other $\in$	$[-1.5,\ -5 \cdot 10^{-8}]$	$[-0.14,\ -10^{-8}]$

Table 3: Distribution of real part of the spectrum of the Jacobian for CBM-IV problems B and C.

Problem	D (urban)	E (rural)
Species i	$\mathrm{Re}(\lambda_i)$	$\mathrm{Re}(\lambda_i)$
OH	-28	-1.81
CRO_2	-1.45	-1.38
CHO_2	-1.45	-1.30
$MAOO$	-1.23	-1.17
$MVKO$	-1.23	-1.17
$MCRG$	-1.23	-1.17
All other $\in$	$[-0.3,\ +1 \cdot 10^{-6}]$	$[-0.03,\ +2 \cdot 10^{-8}]$

Table 4: Distribution of real part of the spectrum of the Jacobian for AL problems D and E.

photolytic reactions. The mechanism, based on the work of Atkinson et. al. [1] and Lurmann et al. [17] can be used to study the chemistry of both highly polluted (e.g., near urban centers) and remote (e.g., marine) environments. Problem D is an urban scenario, while problem E a rural one, based on IPCC scenario 3. The simulated conditions and initial concentrations are identical to those employed in problems B and C, respectively. The values of initial concentrations, and the values of hourly emissions are given in Table 5. The emission was performed in equal quantities at the beginning of each time interval. Information about the stiffness of the models in terms of the eigenvalues of the Jacobian is given in Table 4. Since AL does not treat explicitly $O(^1D)$ and $O(^3P)$, the large negative eigenvalues associated with these species are not present.

2.4 Problem F*: A stratospheric model*

This test problem is based on the chemical mechanism that has been used in the NASA HSRP/AESA stratospheric models intercomparison. The initial concentrations and the values of the rate constants follow the NASA region A scenario[5] with the difference that the photolysis rates were approximated by C^1 functions. There are 34 species[6] involved in 84 thermal and 25 photolytic reactions. The values of initial concentrations for the most important species are given in Table 6. No emissions have been prescribed. For a complete description of the problem we refer to the NASA ftp site. Information about the stiffness of the problem in terms of the eigenvalues of the Jacobian is given in Table 7.

[5]Model available at NASA ftp site, contact Douglas E. Kinnison, kinnison1@llnl.gov.

[6]Plus 6 species whose concentrations were held fixed throughout simulation: H_2O, CO, O_2, H_2, N_2, CH_4.

Problem	B, D (urban)		C, E (rural)	
Species	Initial (ppb)	Emission (ppb/hour)	Initial (ppb)	Emission (ppb/hour)
NO	50	1	0.1	0.01
NO_2	20	0.2	0.1	0.01
$HONO$	1	—	30	—
O_3	100	—	0	—
H_2O_2	1	—	2	—
CO	300	2	100	—
$HCHO$	10	0.2	0	—
ALD_2	10	0.2	0	—
PAN	1	—	0	—
Alkans	50	2	0	—
Alkens	10	1	0	—
Ethene	10	0.2	0	—
Aromatics (AL)	20	0.4	0	—
Toluene (CBM-IV)	10	0.2	0	—
Xylene (CBM-IV)	10	0.2	0	—
Isoprene	10	1	1	0.1
Relative humidity	80%			
Temperature	288.15 K			
Altitude	0 km			
Pressure	1013.25 mbar			
Air density	$2.55 \cdot 10^{19}$ mlc/cm^3			

Table 5: Physical conditions, initial concentrations and hourly emissions for the tropospheric problems B, C, D and E. *Toluene* and *Xylene*, which are treated independently in CBM-IV, are lumped as *Aromatics* in the AL model.

Species	Initial (*ppb*)	Species	Initial (*ppb*)
O	8.15	O_3	656
NO	10.7	NO_2	2.75
HNO_3	0.35	H_2O	6100
OH	0.2	HO_2	0.14
H_2	370	CH_4	490
CO	20	$C\ell O$	1
$HC\ell$	2.15	$HOC\ell$	0.22
Temperature	241.43 K		
Altitude	40 km		
Latitude	65^o N		
Pressure	2.7 mbar		
Air density	$8.12 \cdot 10^{16}$ mlc/cm^3		

Table 6: Initial concentrations and physical conditions for the stratospheric problem F.

Problem	F
Species i	$\mathrm{Re}\,(\lambda_i)$
$O(^1D)$	$-2.53 \cdot 10^6$
HCO	$-1.06 \cdot 10^5$
$C\ell OO$	$-1.70 \cdot 10^4$
CH_3	$-9.98 \cdot 10^2$
H	$-1.17 \cdot 10^2$
CH_3O	-16
$C\ell$	-4.5
$O(^3P)$	-1.37
All other $\in$	$[-0.5,\ +6 \cdot 10^{-8}]$

Table 7: Distribution of real part of the spectrum of the Jacobian for the stratospheric problem F.

Problem	G
Species i	$\mathrm{Re}\,(\lambda_i)$
$HNO_{3\ (aq)}$	$-2.2 \cdot 10^9$
$O(^1D)$	$-8.1 \cdot 10^8$
$OH^-_{(aq)}$	$-1.8 \cdot 10^7$
$SO^{2-}_{3\ (aq)}$	$-1.3 \cdot 10^7$
$HCOOH_{(aq)}$, $SO_{2\ (aq)}$,	-1.25e+7, -3.65e+6, -1e+6,
$HCOO^-_{(aq)}$, $NH_{3\ (aq)}$, $O^{2-}_{(aq)}$,	-4.2e+5, -1.3e+5, -8.2e+4,
$HO_{2\ (aq)}$, OH, O, $H^+_{(aq)}$,	-2e+4, -9e+3, -2.46e+3,
$HSO^-_{3\ (aq)}$,$OH_{(aq)}$, CH_3O,	-2.2e+3, -2e+2, -1.5e+2,
$NO^-_{3\ (aq)}$,ROR,$NO_{3\ (aq)}$	-90, -30,-15
All other $\in$	$[-10,\ +8 \cdot 10^{-7}]$

Table 8: Distribution of real part of the spectrum of the Jacobian for the wet problem G.

2.5 Problem G*: A wet model*

From the numerical point of view, this test problem is the most difficult one. It contains 65 species[7] involved in 77 thermal and 11 photolytic gas-phase chemical reactions, 39 liquid-phase chemical reactions and 39 gas-liquid mass transfer reactions. The gas-phase mechanism is based on CBM-IV, while the liquid-phase mechanism is based on a chemical scheme the authors obtained from [18]. Initial concentrations are given in Table 9. Information about the stiffness of the problem in terms of the eigenvalues of the Jacobian is given in Table 8. Of numerical interest is the fact that only for the four most negative eigenvalues does the relation $\lambda_i \approx -L_i$ hold, while (different from the gas-phase-only test problems) the number of stiff eigenvalues is much larger. This is due to the rapid gas-liquid phase kinetics. In Table 8 we have listed these large negative eigenvalues and the species with large L_i, but without making a one-to-one correspondence between them.

3. The solvers

We have chosen nine solvers from the literature. Four of them are off-the-shelf, general purpose, implicit stiff ODE solvers. Three of them have been modified by implementing a sparse matrix technique, while the fourth already contained such a technique. The other five solvers are explicit and

[7] Plus 5 species whose concentrations were held fixed: H_2O (vapour), H_2O (drops), CH_4, O_2, CO_2(aq).

Species	Initial (ppb)	Emission (ppb/hour)	Species	Initial (ppb)
NO	0.2	0.01	O_3	60
NO_2	0.5	0.01	CO	200
H_2O_2	1.5	—	$HCHO$	1.0
PAR	1.2	—	$Ethene$	$2.4 \cdot 10^{-2}$
$ISOP$	1.0	0.05	$Xylene$	$2 \cdot 10^{-2}$
SO_2	$3.3 \cdot 10^{-2}$	—	$Toluene$	$3 \cdot 10^{-2}$
$Temperature$			288.15 K	
$Altitude$			0 km	
$Relative\ humidity$			80 %	
$Liquid\ water$			0.0436 %	
$Air\ density$			$2.50 \cdot 10^{19}$ mlc/cm^3	

Table 9: Initial concentrations and emissions for the wet problem G.

special purpose. The implicit solvers are fully documented elsewhere and will be discussed here only very briefly. The explicit solvers are not standard. For these we will therefore give the underlying integration formulas. All solvers use variable step size control. To save space, implementation details on the variable step size control are omitted. The interested reader is referred to [9], from where for each solver and each driver program used our Fortran code can be obtained. Before briefly describing the solvers, several general remarks are in order:

- The explicit solvers are applied to the given ODE system (1.1) without problem dependent modifications. Modifications, such as lumping, can improve their performance notably (see e.g. [25] where this is illustrated for the EUSMOG problem), but have the disadvantage of being problem dependent. Hence our comparison emphasizes the numerical properties and performance of the solvers applied.

- The variable step size control requires the choice of a relative error tolerance *rtol* and an absolute error tolerance *atol*. The choices made for *atol* and *rtol* differ per solver and are not specified here. Noteworthy is that at certain times the concentrations of some species (e.g. radicals) can become smaller than 1.0. Because these values are insignificant, they are ignored in the step size control.

- Often we also prescribe a minimal step size. The use of a minimal step size improves efficiency since extremely small steps can be selected by a variable step size selection scheme. Atmospheric chemistry problems containing photochemical reactions can possess time constants as small as 10^{-8} to 10^{-9} seconds and step size selection mechanisms do signal these. However, these extremely small step sizes are redundant because the minimal time constants of importance for photochemical chemistry models are of the order of seconds and species with a time constant truly smaller almost instantaneously get in their (solution dependent) steady state when they are perturbed. On the other hand, too large lower bounds for the step size can cause convergence and loss of accuracy problems to the numerical solvers.

- All sparse implicit solvers work with the analytical Jacobian matrix and can be shown to mimic conservation rules which exist for the ODE system. This does not hold for all of the explicit solvers.

- A numerical comparison should focus on modest accuracies, say relative accuracies near 1%. Higher accuracy levels are redundant for the actual practice of air pollution modelling.

3.1 QSSA

Two QSSA solvers were used. The first is based on the most simple QSSA formula

$$y^{n+1} = e^{-hL(t_n,y^n)}y^n + (I - e^{-hL(t_n,y^n)})L^{-1}(t_n,y^n)P(t_n,y^n), \tag{3.2}$$

where y^n denotes the numerical approximation at time $t = t_n$ and h denotes the step size. This formula is explicit because the loss matrix L is diagonal. The notion explicit thus means that no systems of algebraic equations need to be solved. The second is based on a Richardson extrapolated form of (3.2) and reads [16]

$$\begin{aligned} y_h^{n+1} &= Q(y_n;h), \quad y_{h/2}^{n+1/2} = Q(y_n;h/2), \\ y_{h/2}^{n+1} &= Q(y_{h/2}^{n+1/2};h/2), \quad y^{n+1} = 2y_{h/2}^{n+1} - y_h^{n+1}, \end{aligned} \tag{3.3}$$

where $\mathrm{Q}(y_n;h)$ refers to (3.2). The extrapolated scheme is slightly more than two times expensive per time step of length h. The order of consistency of (3.3) equals two, whereas (3.2) is of order one. See [9] for more details.

3.2 CHEMEQ

One of the first dedicated, explicit methods for solving chemical equations in comprehensive advection-reaction models is the hybrid algorithm of Young and Boris [29]. It is currently implemented in the CALGRID mesoscale model [28]. In the original algorithm, the species are dynamically separated into two categories (fast and slow) according to the relative magnitude of their life-times $\tau_k = 1/L_k$ with respect to the current step size h. Each category is integrated with a special predictor-corrector formula. Our implementation of CHEMEQ follows the one described in [20] and is based on the following predictor-corrector pairs (the abbreviation P_k^n stands for $P_k(t_n,y^n)$, etc.):

- If $\tau_k > 5h$ (slow species):

$$\text{predictor} \quad : \quad y_k^{n+1} = y_k^n + h(P_k^n - L_k^n\, y_k^n) \tag{3.4}$$

$$\text{corrector} \quad : \quad y_k^{n+1} = y_k^n + \frac{h}{2}(P_k^n - L_k^n\, y_k^n + P_k^{n+1} - L_k^{n+1} y_k^{n+1}) \tag{3.5}$$

- If $0.2h \le \tau_k \le 5h$ (intermediate species):

$$\text{predictor} \quad : \quad y_k^{n+1} = \frac{y_k^n\,(2\tau_k^n - h) + 2hP_k^n\,\tau_k^n}{2\tau_k^n + h} \tag{3.6}$$

$$\text{corrector} \quad : \quad y_k^{n+1} = \frac{y_k^n\,(\tau_k^n + \tau_k^{n+1} - h) + \frac{h}{2}\,(P_k^n + P_k^{n+1})\,(\tau_k^n + \tau_k^{n+1})}{\tau_k^n + \tau_k^{n+1} + h} \tag{3.7}$$

- If $\tau_k < 0.2h$ (fast species):

$$\text{steady state assumption} : \quad y_k^{n+1} = \frac{P_k^n}{L_k^n} \tag{3.8}$$

A quick inspection will reveal that the corrector formulas are all derived from the implicit trapezoidal rule. They are applied, however, in an explicit manner. Hence, denoting $y_k^{(i)}$ as the k-th component of the i-th corrected approximation $y^{(i)}$ for y^{n+1}, in all occurrences $y_k^{(i)}$ is simply substituted in the right-hand sides of the corrector formulas, so as to compute the new approximations $y_k^{(i+1)}$. The correctors are applied until

$$\max_k \left| \frac{y_k^{(i+1)} - y_k^{(i)}}{y_k^{(i+1)}} \right| \leq 0.3\ tol,$$

where *tol* is an imposed tolerance value for the step size control which is based on *atol* and *rtol*. In case of non-convergence, the step is rejected and the computations restarted with $h_{new} = 0.6h$. In case of convergence the integration proceeds with $h_{new} = h \min(1.1, fac)$, where $fac = \sqrt{tol/err}$ with *err* the estimation for the local truncation error. It is emphasized that the step selection in the original CHEMEQ is based uniquely on the convergence of corrector iterates, whereas we use an estimation for the local error to govern the step size. However, following CHEMEQ philosophy, a local error greater than *tol* does not force a step rejection. It only restricts h_{new}. See [9] for more details.

3.3 ET

The solver ET uses an extrapolation algorithm proposed in [4]. The approximations used for the extrapolation are computed with a predictor-corrector pair of which the corrector is a QSSA type formula. To describe the formulas used, we adopt the implicit notation used for CHEMEQ. Hence the evaluations of the right-hand side of the implicit formulas have to be thought of as carried out in the same way as for CHEMEQ. The predictor formula implemented in the solver tested in this paper is the simple QSSA formula

$$y^{n+1} = e^{-hL^n} y^n + (I - e^{-hL^n})(L^n)^{-1} P^n. \tag{3.9}$$

Like for CHEMEQ, the corrector dynamically separates the species into three categories:

- If $\tau_k > 100h$, the trapezoidal rule is used,

$$y_k^{n+1} = y_k^n + \frac{h}{2}(P_k^n - L_k^n\, y_k^n + P_k^{n+1} - L_k^{n+1} y_k^{n+1}). \tag{3.10}$$

- If $0.1h \leq \tau_k \leq 100h$, the equations are corrected using the QSSA type formula

$$y_k^{n+1} = \psi_k^{n+1} + (y_k^n - \psi_k^{n+1}) \exp\left[-\left(\frac{1}{L_k^n} + \frac{1}{L_k^{n+1}} \right) \frac{h}{2} \right], \tag{3.11}$$

  where ψ_k^{n+1} is defined by

$$\psi_k^{n+1} = \frac{1}{4} \left(P_k^{n+1} + P_k^n \right) \left(\frac{1}{L_k^{n+1}} + \frac{1}{L_k^n} \right). \tag{3.12}$$

- If $\tau_k < 0.1h$, the steady state assumption is made, i,e.,

$$y_k^{n+1} = \psi_k^{n+1}. \tag{3.13}$$

For details about how the extrapolation is organized and the corrector is used we refer to [4]. See also [9] for more details.

3.4 TWOSTEP

TWOSTEP [24, 25, 26] is based on the variable step size, two-step backward differentiation formula (BDF)

$$y^{n+1} = Y^n + \gamma h f(t_{n+1}, y^{n+1}), \qquad h = t_{n+1} - t_n, \tag{3.14}$$

where $\gamma = (c+1)/(c+2)$, $c = (t_n - t_{n-1})/(t_{n+1} - t_n)$ and

$$Y^n = \left((c+1)^2 y^n - y^{n-1}\right)/(c^2+2c).$$

The 2nd-order BDF formula has been chosen in view of the modest accuracy requirement. Rather than using the common modified Newton iteration, the classical Gauss-Seidel or Jacobi iteration technique is used for computing an approximation to the implicitly defined y^{n+1}. In the application of these techniques we exploit, to some extent, the production-loss form (1.1), by which (3.14) can be written as

$$y^{n+1} = F(y^{n+1}) := \left(I + \gamma h L(t_{n+1}, y^{n+1})\right)^{-1} \left(Y^n + \gamma h P(t_{n+1}, y^{n+1})\right). \tag{3.15}$$

The Gauss-Seidel technique is now applied to the nonlinear system of equations $y = F(y)$. That is, given the iterate $y^{(i)}$ as the i-th approximation for the sought solution y^{n+1}, TWOSTEP computes the next iterate $y^{(i+1)}$ by the componentwise formula

$$y_k^{(i+1)} = F_k\left(y_1^{(i+1)}, \ldots, y_{k-1}^{(i+1)}, y_k^{(i)}, \ldots, y_m^{(i)}\right), \qquad k = 1, \ldots, m. \tag{3.16}$$

This results in an explicit computation owing to the diagonal form of L. More precisely, for the computation of $y_k^{(i+1)}$ only division by the scalar variable $1 + \gamma h L_k(t_{n+1}, v)$ is required, where v denotes the intermediate vector

$$v = [y_1^{(i+1)}, \ldots, y_{k-1}^{(i+1)}, y_k^{(i)}, \ldots, y_m^{(i)}]^T.$$

The fact that in v the first $(k-1)$ components are taken from the new iterate, makes (3.16) a Gauss-Seidel type iteration process. When we take all m components in v from the old iterate $y^{(i)}$, then an iteration method of Jacobi or Picard type results. Computationally there is not much difference between the two, although Gauss-Seidel has to be programmed in line. Hence Jacobi iteration is somewhat easier to use. Here we will use both types of iteration techniques. Note that for the Gauss-Seidel technique the order of the species plays a role when only a few number of iterations are used. In all experiments we in fact restrict the number of iterations to only two. In [25, 26] this has been shown to work well. For more details on the code we refer to [25, 26] and [9].

3.5 VODE

VODE is a "Variable coefficient Ordinary Differential Equation" solver based on the implicit BDF formulas [2, 11] and a successor of the "Livermore Solver" LSODE from [14]. The latter is popular in the field of atmospheric chemistry. For a discussion of the mathematical techniques implemented we refer to [2, 11]. We used VODE as a black box with its user parameter $istart = 1$, except that we modified the code to carefully exploit the sparsity of the Jacobian matrix. This reduces the costs of solving the linear algebraic systems arising in the modified Newton iteration. In [15] and [25] it has been shown that this is very profitable for atmospheric chemistry problems. We used the sparse linear algebra implementation described in [19]. The necessary routines are automatically generated by the symbolic chemical preprocessor KPP [5], which transparently:

- determines the sparse analytical Jacobian,
- reorders the species using a diagonal Markowitz criterion, in order to minimize the fill-in resulting from the LU decomposition of the matrix used in the modified Newton process,
- analyses the pattern of zeros in the Jacobian and builds the data structures needed for the sparse Doolitle LU decomposition,
- generates loop-free code for the forward-backward substitution routines.

The performance of VODE appeared to be sensitive to the choice of the absolute tolerances. Using the natural value $atol = 1.0$ was not always optimal. We therefore set them componentwise as

$$atol_i = \max\left(10^{-2}, 10^{-2} \cdot rtol \cdot \eta_i\right) \quad (\text{mlc/cm}^3)$$

where η_i estimates the magnitude of the concentration of species i. See [9] for other specific parameter settings.

3.6 SDIRK4

This solver has been borrowed from Hairer and Wanner [11] where a full description along with numerical results can be found. It is based on a 4-th order, diagonally implicit Runge-Kutta method using five stages. Because this solver is of one-step type, it allows a fast increase in step size after a restart. For atmospheric chemistry applications this is an obvious advantage. We have only modified it for the treatment of sparsity as described in Section 3.5. Hence all strategies were unaltered. See [9] for specific parameter settings.

3.7 RODAS

This solver has also been borrowed from Hairer and Wanner [11]. It is based on a 4-th order, Runge-Kutta-Rosenbrock method using six stages. This solver is also of one-step type and hence shares the advantage of a fast increase in step size after a restart with SDIRK4. The code has been modified for the treatment of sparsity as described in Section 3.5. All strategies were unaltered. See [9] for specific parameter settings.

3.8 LSODES

LSODES is a version of the popular BDF code LSODE which exploits the sparsity in the Jacobian matrix by calling linear algebra routines from the Yale Sparse Matrix Package [7, 6]. It is obvious that VODE and LSODES are closely related. For our application an important difference is that VODE uses a dedicated sparsity technique, whereas LSODES uses the general Yale package, which is less efficient, in general. LSODE and LSODES are often used to solve atmospheric chemical kinetics equations (see e.g. [20]). The code was applied with its user parameter setting MF = 121, i.e. analytical Jacobian with an inner estimation of the sparsity structure. See [9] for other specific parameter settings.

4. SETUP OF EXPERIMENTS

The solvers are tested as if they were used in an operator splitting environment. This means that we split the total integration interval into N subintervals of length Δt, each representing a step taken by the advection scheme in the assumed operator splitting. For each subinterval we then restart the integration of the stiff solvers. For all test problems the length of the subintervals equals $\Delta t = 3600$ sec. leading to $N = 120$ new starts over the 5 day period.

All test problems contain photochemical reactions. This means that part of the reaction constants are determined by time of the day dependent photolysis rates which undergo a rapid change at sunrise and sunset. This change gives rise to large variations in concentration values and normally force a solver to adjust the step size. In Problem `A` the photolysis rates are given by a C^0 function, while in Problems `B, C, D, E, F` and `G` by a C^1 function which are zero at nightly periods.

All the runs were made in double precision on a HP-UX 935 A workstation with a CPU clock frequency of 125 megaherz and 160 Mbytes RAM. The numerical results for all test problems are compared to a very accurate reference solution computed by the code RADAU5 from [11] with the very tight tolerances $rtol = 10^{-12}$, $atol_i = 10^{-15}\eta_i$, where η_i estimates the magnitude of the concentration of species i in unit mlc/cm^3. Our measure of accuracy is based on a modified root mean square norm of the relative error. With the reference solution y and the numerical solution $\hat{y}$ available at $\{t_n = t_0 + n\Delta t,\ 0 \le n \le N\}$, we first compute for each species k

$$ER_k = \sqrt{\frac{1}{|\mathcal{J}_k|} \cdot \sum_{n \in \mathcal{J}_k} \left| \frac{y_k(t_n) - \hat{y_k}(t_n)}{y_k(t_n)} \right|^2}, \quad \text{where} \quad \mathcal{J}_k = \{0 \le n \le N : \ |y_k(t_n)| \ge a\} \ . \qquad (4.17)$$

Hence we compute specieswise a temporal error measure, which we then represent in the plots in two ways. Through the number of significant digits for the average of ER_k, defined by

$$SDA_1 = -\log_{10}\left(\frac{1}{m}\sum_{k=1}^{m} ER_k\right), \tag{4.18}$$

and the number of significant digits for the maximum of ER_k, defined by

$$SDA_\infty = -\log_{10}(\max_k ER_k). \tag{4.19}$$

The threshold factor a used here is given the value $a = 1.0$. If the set $\mathcal{J}_k$ is empty, the value of ER_k is neglected. The purpose of considering the above defined error measure instead of the root mean square norm ($a = 0$) is to avoid chemically meaningless large relative errors for concentration values smaller than 1.0 mlc/cm^3. It is instructive to present the accuracy of the computed results using averaged and maximal errors taken over the number of species. Finally, in the work-precision diagrams of the following section, efficiency is measured by CPU time. We thus plot the SDA values against the measured CPU times on a logarithmic scale in unit seconds. Observe that $SDA = 2$ means 1% accuracy in the error measure used. In discussing the results presented in the next section we focus on this accuracy level.

5. Results and illustrations

5.1 Problem A*: The* EUSMOG *model*

The work-precision diagram is given in Figure 1. The implicit integrators and the two versions of TWOSTEP perform very well. For an accuracy of two digits, sparse VODE appears to be the fastest. The good performance of implicit integrators is due in part to the small dimension of the system which means less work with the linear algebra. Note that the JACOBI and (GAUSS-)SEIDEL versions of TWOSTEP have similar performance for this test problem (see also [25]). It is clear that the simplest QSSA solver, ET and CHEMEQ are the slowest among the tested routines. It is also worth noting that the QSSA performance greatly improves by extrapolation.

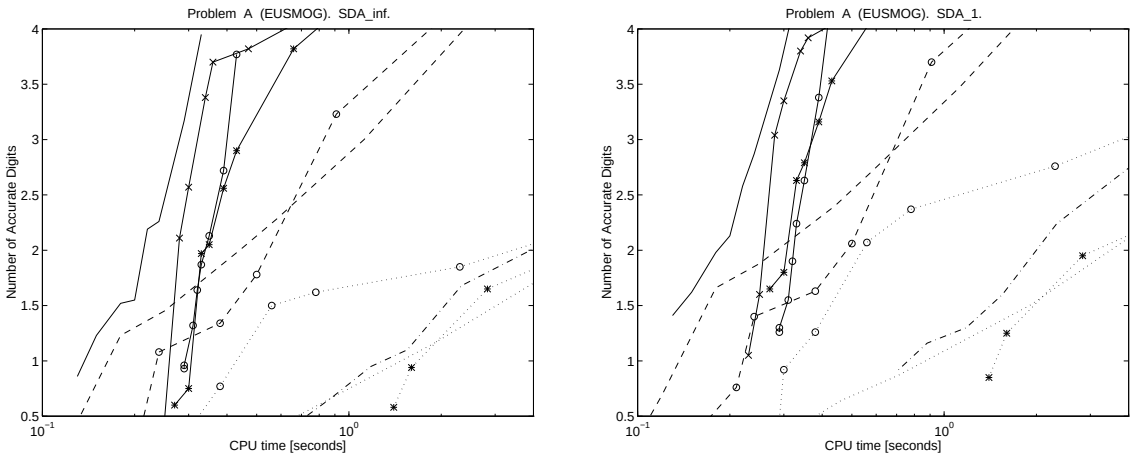

Figure 1: Work-precision diagram for test problem A (EUSMOG): TWOSTEP SEIDEL (dashed), TWOSTEP JACOBI (dashed with "o"), QSSA (dots), Extrapolated QSSA (dots with "o"), ET (dots with "$*$"), CHEMEQ (dash-dots), Sparse VODE (solid), Sparse SDIRK4 (solid with "$*$"), Sparse RODAS (solid with "x") and LSODES (solid with "o").

5.2 Problems B *and* C*: The* CBM-IV *model*

In Figure 2 the numerical results for test problems B and C are presented. For obtaining two accurate digits, RODAS appears to be the fastest, followed by SDIRK, VODE, LSODES and TWOSTEP SEIDEL. The latter is the best when less accuracy is demanded, while the implicit codes are preferable for higher precisions. This is due to the fact that they use higher order formulas (nicely represented by the higher slopes of their diagrams). An interesting remark is that the slope of the TWOSTEP JACOBI diagram decreases when higher accuracies are required. This is due to an imposed minimal step size of 0.01 sec., which makes the convergence of the Jacobi iteration very slow on part of the time interval. Since TWOSTEP was used with a fixed number of only two iterations, this is directly reflected in the accuracy of the solution. A minimal step size larger than 0.1 sec. creates similar problems in TWOSTEP SEIDEL. The gap between the two TWOSTEP diagrams is due primarily to the different values of the minimal step size used: 0.01 sec. for JACOBI and 0.1 sec. for SEIDEL. The explicit solver ET is not competitive. It needs 2 minutes CPU time to integrate problem B ($\text{SDA}_1 = 1.3$) and 3.5 minutes to integrate problem C ($\text{SDA}_1 = 1.1$). Thus, its work-precision diagram is situated to the far right of Figure 2 and is not plotted. Among the other integrators, QSSA is clearly the slowest, but by extrapolation it gains about one accurate digit for the same CPU time. In both scenarios CHEMEQ performs better than the plain QSSA scheme but worse than Extrapolated QSSA. As expected, the LSODES and sparse VODE diagrams are similar, except that the LSODES one is shifted to the right. Working with predefined sparsity data structures, VODE is consistently faster than the general purpose LSODES. However, despite its generality, for test problems B and C LSODES performs very well.

5.3 Problems D *and* E*: The* AL *model*

The results are reported in Figure 3. It is interesting to compare code performances to those obtained for test problems B and C, since the same urban and rural scenarios are simulated with both CBM-IV and AL. They differ however in the number of reactions and species, the current problems D and E being much larger. If a standard implementation of the implicit solvers was used, their linear algebra workload would have increased as m^3, with m the number of species, while for dedicated explicit integrators the workload increases linearly with m. Thus, at first sight, for sufficiently large problems use of explicit integrators seems to be preferable. Because we use a sparse linear algebra implementation, the situation becomes truly different. For the sparse implementation a rough estimation of the linear algebra workload is given by the number of nonzero elements in the Newton matrix. As seen in Table 1, this number increases almost linearly with m for the test mechanisms considered here. This means that even for fairly large chemical systems, sparse implicit solvers may very well remain competitive. Our test results shown in Figure 3 clearly illustrate this. For both problems all sparse implicit solvers outperform the dedicated explicit ones, with the exception of TWOSTEP SEIDEL. The gap between this code and its JACOBI version is again due to the fact that SEIDEL iterations allow the use of larger values for the minimal step size. Still, this imposed minimal step size causes convergence problems at part of the time interval, which explains the curious slopes of TWOSTEP in the range of high accuracies for the more difficult urban scenario. Noteworthy is that the one-step solver RODAS is the fastest in the 1% error region. For the more difficult urban problem, the two one-step solvers RODAS and SDIRK4 are always faster than their BDF counterparts VODE and LSODES. The explicit solver ET fails to integrate problem D and is among the slowest for problem E. The CHEMEQ diagram is in between that for QSSA and EXTRAPOLATED QSSA. The latter clearly performs better in the rural cases than in the urban ones for both CBM-IV and AL mechanisms.

5.4 Problem F*: The stratospheric model*

This problem has about the same dimension as the two CBM-IV problems, but the integrators perform quite differently relative to each other as shown in Figure 4. The implicit integrators work best. There is not much difference between their performances, but there is a large gap between them and the explicit codes, with the largest for ET, CHEMEQ and the two QSSA solvers. In particular, for this problem the implicit codes require less CPU time than for CBM-IV, whereas for TWOSTEP the amount

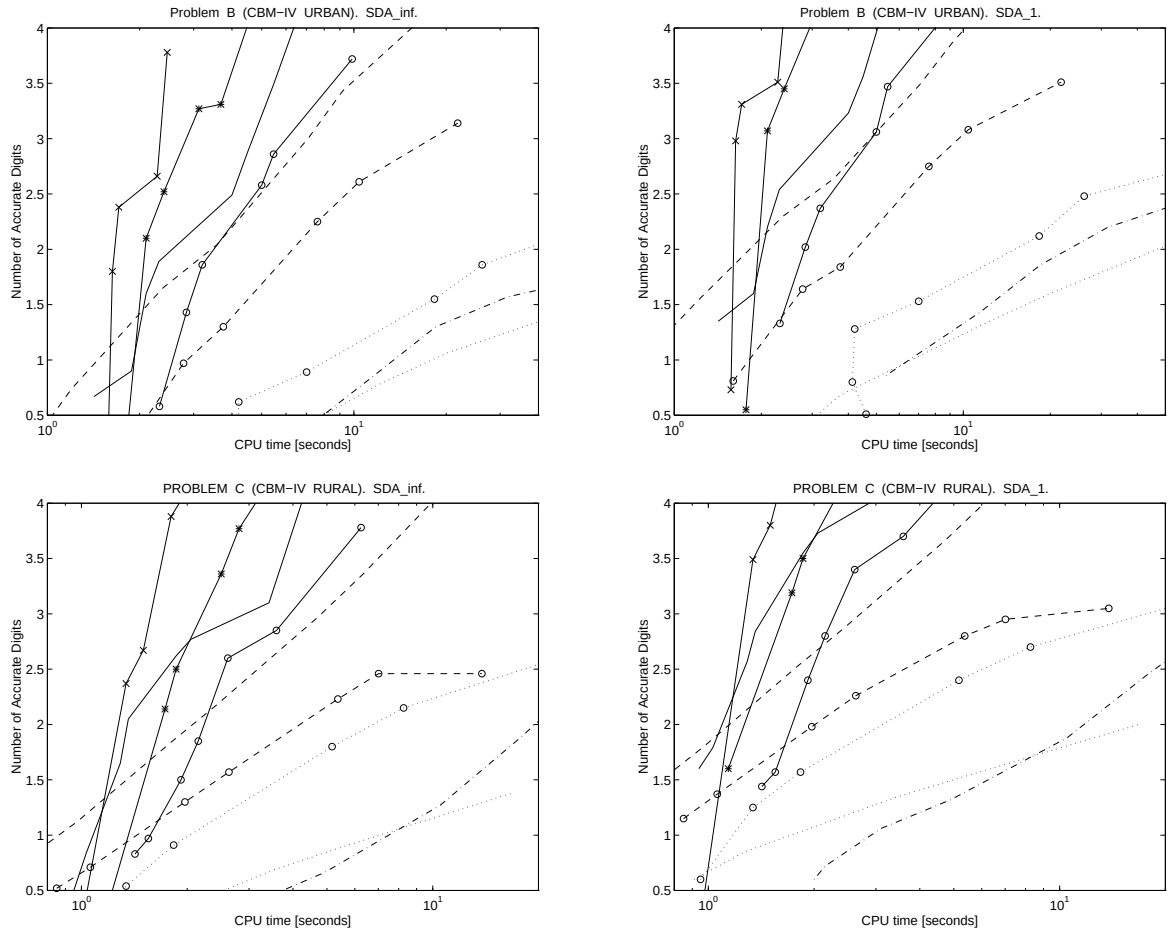

Figure 2: Work-precision diagram for test problems `B` and `C` (CBM-IV): The upper pair of diagrams correspond to problem `B` and the lower pair to problem `C`. TWOSTEP SEIDEL (dashed), TWOSTEP JACOBI (dashed with "*o*"), QSSA (dots), Extrapolated QSSA (dots with "*o*"), CHEMEQ (dash-dots), Sparse VODE (solid), Sparse SDIRK4 (solid with "∗"), Sparse RODAS (solid with "x") and LSODES (solid with "*o*").

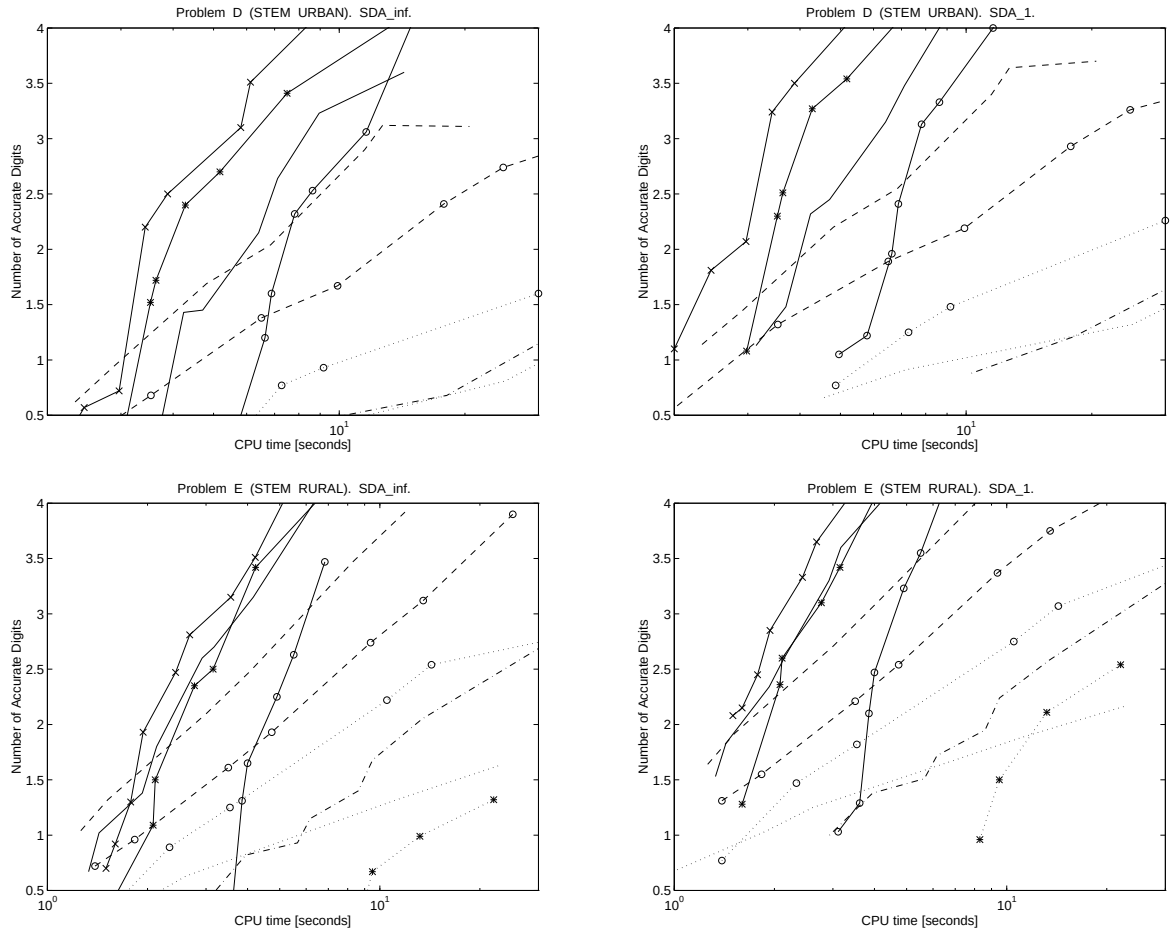

Figure 3: Work-precision diagram for test problems `D` and `E` (AL): The upper pair of diagrams correspond to test problem `D`, while lower pair to test problem `E`. TWOSTEP SEIDEL (dashed), TWOSTEP JACOBI (dashed with "*o*"), QSSA (dots), Extrapolated QSSA (dots with "*o*"), ET (dots with "∗"), CHEMEQ (dash-dots), Sparse VODE (solid), Sparse SDIRK4 (solid with "∗"), Sparse RODAS (solid with "x") and LSODES (solid with "*o*").

of CPU time almost remains equal. From an additional investigation we learned that the absence of emissions in the stratospheric problem must play a role here. Without emissions, concentration values vary less in between the one hour subintervals. The implicit solvers enjoy this, since as a rule they allow larger step sizes than TWOSTEP. We have checked this observation by removing the emissions in the CBM-IV model. In this case the implicit solvers also integrate much faster, the CPU timings being then similar to those obtained for the stratospheric problem.

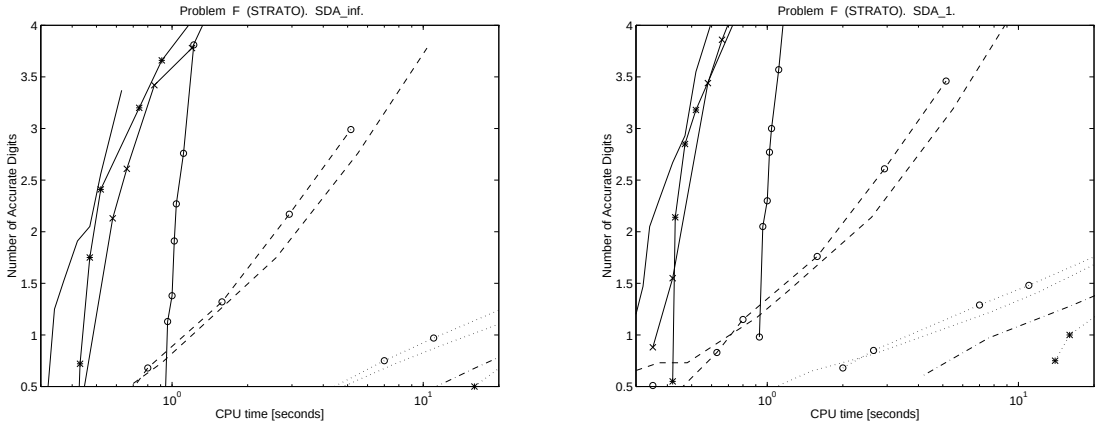

Figure 4: Work-precision diagram for test problem F (STRATO): TWOSTEP SEIDEL (dashed), TWOSTEP JACOBI (dashed with "o"), QSSA (dots), Extrapolated QSSA (dots with "o"), ET (dots with "$*$"), CHEMEQ (dash-dots), Sparse VODE (solid), Sparse SDIRK4 (solid with "$*$"), Sparse RODAS (solid with "x") and LSODES (solid with "o").

5.5 Problem G*: The wet model*

As we have previously seen, this test problem has a large number of stiff eigenvalues, most of which cannot be associated to certain short lived species. This makes the test problem G numerically challenging in the sense that the explicit methods fail. Numerical results confirm that the explicit formulas described in Section 3 cannot solve this problem with a reasonable efficiency. We tested each routine with the loosest restrictions on step size and tolerance for which the numerical results were still meaningfull. See the codes [9] for the exact setting of the parameters. The timings obtained for integrating the first 10 seconds (or 1 hour) of evolution are given in Table 10. When looking at the results, keep in mind the fact that all the implicit solvers integrate the test problem along the five days interval in less than 10 seconds CPU time. Among the dedicated integrators, TWOSTEP SEIDEL performs best. However, even this code selects very small step sizes, which is not the case when a fully implicit implementation of the BDF2 formula is used (VODE with restricted maximal order). The behaviour of the dedicated integrators is typical for standard explicit formulas applied to general stiff problems. To make the terms of comparison more clear, we estimated the CPU time that would be needed to complete the five days simulation. This estimated time is given in the last column of Table 10. The explicit solver ET gives a floating point exception on this test problem and is not included in the table.

Some effort has been made to optimize the performance of the implicit solvers for this test problem. RODAS and SDIRK4 were used with a minimal stepsize of 10^{-3} sec., while for VODE a minimum of 10^{-8} sec. appeared to be best. No minimal step size was prescribed for LSODES. For sparse VODE the maximal order was restricted to 3. Although the original VODE had no problems integrating this model, the sparse version selected tiny step sizes when a maximal order of 4 or 5 was used. The work-precision diagram for the implicit schemes is given in Figure 5, which shows that RODAS, VODE

and SDIRK4 perform equally well.

Integrator	Simulated interval	CPU time	Estimated total CPU
QSSA	10 sec	19 min	350 days
Extrap QSSA	10 sec	33 min	650 days
CHEMEQ	10 sec	0.5 min	15 days
TWOSTEP J	10 sec	9 min	270 days
TWOSTEP S	3600 sec	5.3 min	10.3 hours

Table 10: The timing of dedicated integrators on test problem G. The last column represents the estimated CPU time needed to complete the five days simulation.

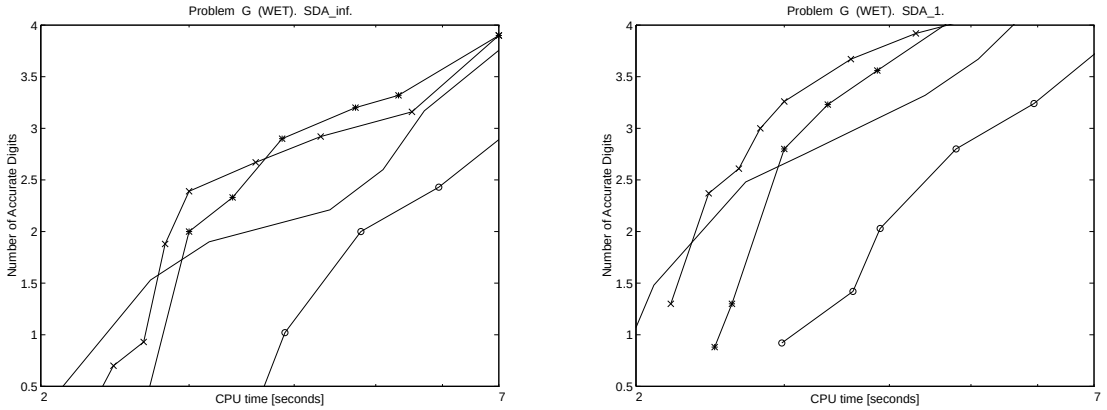

Figure 5: Work-precision diagram for test problem G (WET): Sparse (solid), Sparse SDIRK4 (solid with "∗"), Sparse RODAS (solid with "x") and LSODES (solid with "*o*").

6. Overall conclusions and remarks

Although we have taken any reasonable precaution in implementing the models and in testing the codes, still undiscovered errors and/or less optimal settings of user parameters may have affected part of the numerical results presented here. The interested reader therefore is invited to repeat the experiments using our codes from [9] and to join us [8] in this benchmark activity. In future work we will continue evaluating hybrid and other numerical schemes. The results presented in the present paper indicate to the following conclusions and remarks:

- Of the dedicated explicit solvers, TWOSTEP is clearly the best. This solver outperforms the others on all problems, often with a wide gap. TWOSTEP is advocated for gas-phase problems only. This code should not be applied to gas-liquid phase problems. In general TWOSTEP SEIDEL is more efficient than TWOSTEP JACOBI. However, Gauss-Seidel iteration must be programmed in line which makes TWOSTEP JACOBI somewhat easier to use. It is important to note that dedicated explicit solvers can sometimes be significantly improved by problem dependent modifications like lumping and/or group iteration. Of course, this requires a considerable knowledge of the reaction mechanism.

[8] Contact J.G. Verwer (janv@cwi.nl) and A. Sandu (sandu@cgrer.uiowa.edu).

- The sparse implicit solvers work efficiently for all problems tested here, including the gas-liquid phase one. In all the cases they give the fastest solution, when two or more accurate digits are required. In general RODAS, VODE and SDIRK4 have comparable performances, although their ranking relative to each other may differ per problem. In most test cases sparse RODAS is the fastest in the 1% error region, with VODE being the best for problems `A` and `F`. The one-step Runge-Kutta solvers RODAS and SDIRK4 clearly work better on the urban AL problem. It should be noted that VODE appeared to be somewhat sensitive to the choice of the absolute tolerances and the choice of a minimal step size.

- In most cases sparse VODE is more efficient than the related sparse BDF solver LSODES. We owe this to the fact that VODE uses a dedicated sparsity technique, whereas LSODES uses the general Yale sparse matrix package. We should also mention that LSODES is used without a precribed minimal stepsize. Some additional runs with sparse VODE without a minimal step size as well, demonstrated that this plays a minor role, the difference in sparse matrix treatment being more important. Both have been applied with the extra storage option for the Jacobian matrix so as to avoid Jacobian updates when possible.

- Since all the tested implicit solvers are general purpose ones, except for the use of sparsity, we definitely think that within this class of methods there is considerable room for efficiency improvements for atmospheric chemistry applications. The current research will therefore be continued with a search to determine also a best or near-to-best sparse implicit solver for atmospheric chemistry problems.

- The best sparse implicit solvers and the best explicit solver (TWOSTEP) should also be compared in 3D model applications. While box model tests are needed to select and develop promising ODE solvers, in real 3D transport-chemistry models other factors should be taken into account as well. Quite important is the length of the time step in the operator splitting, since this determines the number of restarts. Restarts are expensive for implicit codes and one-step methods of Runge-Kutta type have an advantage here over multistep methods. Also robustness and ease of use are important in 3D models, since a subtle tuning of the ODE code is cumbersome due to the large variety of conditions that will occur at different gridpoints. Finally, the issues of memory use, vectorization [15, 27] and parallellization are of great practical importance too. Optimal ODE solvers should be tested in a 3D software environment where vectorization and parallellization take place.

Acknowledgements The work of the authors Sandu, Carmichael and Potra was supported in part by the DOE under grant no DE-FG02-94ER 61855 and by the Center for Global and Regional Environmental Research. The authors Verwer and Van Loon gratefully acknowledge support from the RIVM (Dutch National Institute of Public Health and Environmental Protection) for the projects EUSMOG and CIRK.

REFERENCES

1. R.D. Atkinson, D.L. Baulch, R.A. Cox, R.F.JR. Hampson, J.A. Kerr, and J. Troe. Evaluated kinetic and photochemical data for atmospheric chemistry. *Journal of Chemical Kinetics*, 21:115–190, 1989.
2. P.N. Brown, G.D. Byrne, and A.C. Hindmarsh. VODE: A variable coefficient ODE Solver. *SIAM J. Sci. Stat. Comput.*, 10:1038–1051, 1989.
3. G.R. Carmichael, L.K. Peters, and T. Kitada. A second generation model for regional-scale transport/chemistry/deposition. *Atmospheric Environment*, 20:173–188, 1986.

4. D. Dabdub and J.H. Seinfeld. Extrapolation techniques used in the solution of stiff ODEs associated with chemical kinetics of air quality models. *Atmospheric Environment*, 29:403–410, 1995.

5. V. Damian-Iordache and A. Sandu. KPP - A symbolic preprocessor for chemistry kinetics - User's guide. Technical report xx, University of Iowa, Department of Mathematics, 1995.

6. S.C. Eisenstat, M.C. Gursky, M.H. Schultz, and A.H. Sherman. Yale Sparse Matrix Package. ii. The nonsymmetric codes. Research Report 114, Department of Computer Science, Yale University, 1977.

7. S.C. Eisenstat, M.C. Gursky, M.H. Schultz, and A.H. Sherman. Yale Sparse Matrix Package. i. The symmetric codes. *Int. J. Num. Meth. Eng.*, 18:1145–1151, 1982.

8. M. Prather et. al. Intercomparison of tropospheric chemistry/transport models. *Scientific assesment of ozone depletion, World meteorological organization*, 1995.

9. ftp.cgrer.uiowa.edu. Ftp site at the Center for Global and Regional Environmental Research, University of Iowa (Instructions in directory outgoing/Benchmark), 1996.

10. M.W. Gery, G.Z. Whitten, J.P. Killus, and M.C. Dodge. A photochemical kinetics mechanism for urban and regional scale computer modeling. *Journal of Geophysical Research*, 94:12925–12956, 1989.

11. E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Springer-Verlag, Berlin, 1991.

12. Ø. Hov, I. Isaksen, and E. Hesstvedt. A numerical method to predict secondary air pollutants with an application on oxidant generation in an urban atmosphere. *WMO publication*, 510:219–226, 1978.

13. E. Hesstvedt, Ø. Hov, and I. Isaksen. Quasi-steady-state-approximation in air pollution modelling: comparison of two numerical schemes for oxidant prediction. *Int. J. Chem. Kinet.*, 10:971–994, 1978.

14. A.C. Hindmarsh. *ODEPACK: A systematized collection of ODE solvers.* In: R.S. Stepleman (ed.), IMACS Trans. on Scientific Computation, Vol. 1, Scientific Computing, North Holland, Amsterdam, 1983.

15. M.Z. Jacobson and R.P. Turco. SMVGEAR: a sparse-matrix, vectorized Gear code for atmospheric models. *Atmospheric Environment*, 17:273–284, 1994.

16. L.O. Jay, A. Sandu, F.A. Potra, and G.R. Carmichael. Improved QSSA methods for atmospheric chemistry integration. Technical report 67, University of Iowa, Department of Mathematics, 1995.

17. F.W. Lurmann, A.C. Loyd, and R. Atkinson. A chemical mechanism for use in long-range transport/acid deposition computer modeling. *Journal of Geophysical Research*, 91:10,905–10,936, 1986.

18. J. Matthijssen. Private communication, Laboratoire d'Aerologie OMP, Toulouse, France, 1995.

19. A. Sandu, F.A. Potra, V. Damian, and G.R. Carmichael. Efficient implementation of fully implicit methods for atmospheric chemistry. Technical report 79, University of Iowa, Department of Mathematics, 1995.

20. R. D. Saylor and G. D. Ford. On the comparison of numerical methods for the integration of kinetic equations in atmospheric chemistry and transport models. *Atmospheric Environment*, 29:2585–2593, 1995.

21. D. Simpson, Y. Andersson-Sköld, and M.E. Jenkin. Updating the chemical scheme for the EMEP MSC-W oxidant model: current status. Technical Report 2/93, EMEP MSC-W, The Norwegian Meteorological Institute, Oslo, 1993.

22. M. van Loon. Numerical smog prediction I: The physical and chemical model. CWI Report NM-R9411, Center for Mathematics and Computer Science, Amsterdam, 1995.

23. M. van Loon. Numerical smog prediction II: Grid refinement and its application to the Dutch smog prediction model. CWI Report NM-R9523, Center for Mathematics and Computer Science, Amsterdam, 1995.

24. J.G. Verwer. Gauss-Seidel iterations for stiff ODEs from chemical kinetics. *SIAM Journal of Scientific Computing*, 15:1243–1250, 1994.

25. J.G. Verwer, J.G. Blom, M. van Loon, and E.J. Spee. A comparison of stiff ODE solvers for atmospheric chemistry problems. *Atmospheric Environment*, 30:49-58, 1996.

26. J.G. Verwer and D. Simpson. Explicit Methods for stiff ODEs from atmospheric chemistry. *Applied Numerical Mathematics*, 18:413-430, 1995.

27. J.G. Verwer, J.G. Blom and W.H. Hundsdorfer. An implicit-explicit approach for atmospheric transport-chemistry problems. *Applied Numerical Mathematics* (to appear in Vol. 20, No. 1) 1996.

28. R. Yamartino, J. Scire, G.R. Carmichael, and Y.S. Chang. The CALGRID mesoscale photochemical grid model. *Atmospheric Environment*, 26 A:1493–1512, 1992.

29. T. R. Young and J. P. Boris. A numerical technique for solving stiff ODEs associated with the chemical kinetics of reactive flow problems. *Journal of Physical Chemistry*, 81:2424–2427, 1977.